A Stacked Denoising Autoencoders Based Collaborative Approach for Recommender System

Baojun Niu, Dongsheng Zou(✉), and Yafeng Niu

Chongqing University, Chongqing 400044, China
dszou@cqu.edu.cn

Abstract. This paper uses an autoencoder neural network as user feature learning component for collaborative filtering task. We propose a stacked denoising autoencoder (SDAE) based model to alleviate the sparseness issues in recommendation system. Our model also extends the scalability of CF-based methods in the Top-N recommendation task. Experiments on MovieLens datasets and the result confirmed the effectiveness and potential of our model.

Keywords: Collaborative filtering · Recommender system · Stacked denoising autoencoder

1 Introduction

Recommender System (RS) is an effective tool to deal with information overload problem [1]. As the main thrust in the field of RS research, collaborative filtering(CF) methods are widely used for its knowledge domain free. CF-based approaches can be divided into two branches: neighborhood-based and model-based [2]. Neighborhood-based approaches focus on spotting the similarity relationships among either users or items. Herlocker et al. [3] carried out a comprehensive research into user-based collaborative algorithms. Badrul et al. [4] presented the item-based collaborative system to pre-compute the item-item similarities.

In contrast to the neighbor-based method, singular value decomposition (SVD) [6] is the most prevalent model-based method for Matrix factorization (MF) [5]. SVD performs well in building accurate models for RS by applying elaborated and succinct algebraic structure. Nevertheless, SVD method is only well-defined when the matrix is complete, still facing the problem of sparseness. Imputation [7] is a technique to get a relatively denser matrix filled with baseline estimations for missing ratings. However, inaccurate imputation might distort the original data considerably.

In recent years, research in the neural network has taken a breakthrough in deep layer architecture. The deep belief networks (DBN) [8] inspire researchers to integrate neural network with CF for patterns learning or useful features extracting [9] uses Restricted Boltzmann Machines (RBM) to perform CF, and [10] extended this work to a noo-iid framework. Autoencoder (AE) as a special neural network has achieved good performance in the text processing, image classification tasks. In this paper, we employ

G. Chen et al. (Eds.): PAAP 2017, CCIS 729, pp. 172–181, 2017.
DOI: 10.1007/978-981-10-6442-5_15

AE as a user preference learning component in collaborative filtering and develop a stacked denoising autoencoder (SDAE) based model to alleviate sparseness issue in RS.

2 Traditional Autoencoder

A traditional autoencoder is a feedforward neural network illustrated in Fig. 1. AE can learning representations of the raw data by reconstructing the input data from its output. Usually, it has a three-layered structure, the input and the output has the same number of neurones, and a hidden layer is in the middle.

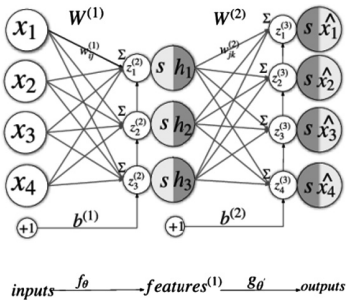

Fig. 1. Traditional autoencoder

Training an autoencoder is to minimise the reconstruction error, solve the following optimisation problem:

$$\begin{aligned} L(x,\hat{x}) &= \frac{1}{2m}\sum_{i=1}^{m}\left\|\hat{x}^i - x^i\right\|^2 + \frac{\lambda}{2}\left(\left\|W^{(1)}\right\|^2 + \left\|W^{(2)}\right\|^2\right) \\ \underset{\theta,\theta'}{\arg\min}\, L(x,\hat{x}) \end{aligned} \tag{1}$$

To get the hyper-parameters $\mathrm{W}^{(l)}, b^{(l)}, l = 1, 2$ in (1). Stochastic Gradient Descent (SGD) is an efficient method to train autoencoder. The key steps of SGD is computing the partial derivatives. We use backpropagation algorithm to compute these partial derivatives. Steps as follows:

For each training example x^i

1. Feedforward pass the inputs, we get activations for each layer: x^i,h^i, $\hat{x}^i$.
2. Backpropagate the errors:
   (1) Compute the error of output layer,

$$\delta^{(2)} = (\hat{x}^i - x^i) \cdot \hat{x}^i \cdot (1 - x^i) \tag{2}$$

   (2) Backpropagate the output error $\delta^{(2)}$ to the hidden layer,

$$\delta^{(1)} = (W^{(2)})^T \cdot \delta^{(2)} \cdot h^i \cdot (1 - h^i) \tag{3}$$

3. Compute the desired partial derivatives, set as

$$\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)}(x^i)^T, \frac{\partial L}{\partial b^{(1)}} = \delta^{(l)} \tag{4}$$

$$\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)}(h^i)^T, \frac{\partial L}{\partial b^{(2)}} = \delta^{(2)} \tag{5}$$

4. After T rounds iterations when finally the value of (1) tends to be stable, we can take h^i, the high-order vector of original x^i, as a more concentrated and effective feature representation of raw input x^i.

3 Construct SDAE

SDAE stands for Stacked Denoising Autoencoder, which is a variant of traditional autoencoder. We can impose constraints on AE to discover more useful representation. By altering the number of units in the hidden layer, we can get a lossy compressed representation feature. But purely altering the size of bottleneck layers may not guarantee to extract good features consistently. [11] proposed a strategy to corrupt the clean input partially for constructing a denosing autoencoder structure(DAE). There are three commonly used noise models to corrupt the input, the additive Gaussian noise, the salt-and-pepper noise and the Masking noise.

The training process of DAE is the same as an AE, except that it takes corrupted input $\tilde{x}$ generated from a particular noise model and then minimise reconstruction error between its outputs and the original clean inputs. That is, the overall loss function is still the same. Figure 2. shows a schematic for these procedures.

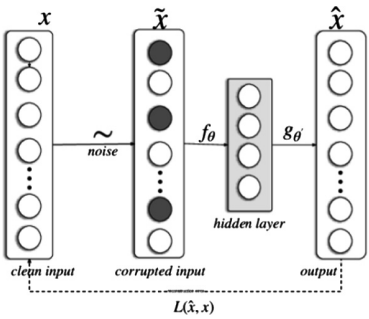

Fig. 2. The schematic of denoising autoencoder

Based on all of the foundations above, learning even higher level representations would be possible by stacking denoising autoencoders. SDAE is a bit deeper neural network constructed with multiple levels of DAEs in which the hidden outputs of each level is accepted as inflow to the successive.

Greedy layer-wise training [12] is an efficient way to obtain parameters for SDAE or SAE. After training the first level autoencoder for corrupted input, its learnt encoding operator f_θ acts on clean input to get the first level hidden layer vector, then use it to train the second level DAE to learn encoder $f_\theta^{(2)}$. From here, repeat the procedure. After training a stack of encoders, perform backpropagation through the whole system to fine-tune the hyper-parameters globally. Figure 3 illustrates the schematic for training the SDAE.

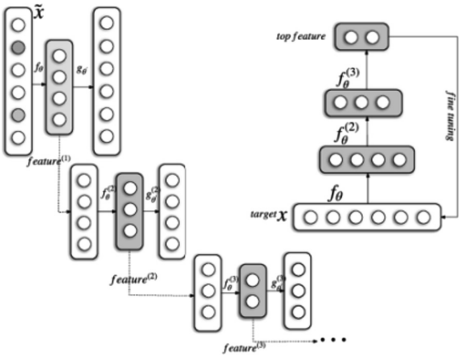

Fig. 3. Schematic for training the SDAE

3.1 SDAE Applied in Collaborative Filtering

In view of the ability of AE to mining inherent structure of data in many domains, we bring SDAE into collaborative filtering, trying to excavate delicate structure of user's preference, and obtain the compressed representation of user's behaviour vector, then apply these high-level features in recommendation task.

We model user's preference behaviour as a vector consisting of ratings over items, denoting $p_{u_i} \in \Re^n$, for example, in a 5-stars scaled rating system for movies. There exist a target user u_t which has a rated items collection $I_{u_t} = \{i_1, i_3, i_4, i_5\}$, in which $r_{t,1} = 5, r_{t,3} = 2, r_{t,4} = 3, r_{t,5} = 4$, then we can get $p_{u_i} = (5, NULL, 2, 3, 4, NULL, \cdots, NULL)^T$.

As shown above, due to the sparseness of rating matrix for RS, the user behaviour preference vector containing a large number of *NULL* value or missing value. There are many reasons for these missing ratings beyond only not liking it, so we can't fill these *NULLs* with "0" and feed this kind of input to autoencoder, otherwise, all these zero ones will be treated as negative preference leading to a strong bias in the system.

To address issue stated above, we extend the user preference vector to a user preference matrix denoting as $p_{u_i} \in \Re^{n \times S}$ where S is the numerical rating scale. In this

$n \times S$ matrix, if exists the rating $r_{i,j}$ on item i_j by user u_i valued with c, then we set the $p^c_{u_i,i_j} = 1$,otherwise 0, and we set the formula as

$$p^c_{u_i,i_j} = \begin{cases} 1, & if \ \ni r_{i,j} = c, \quad i_j \in I_{u_i} \\ 0, & if\, c \neq r_{i,j}, \quad i_j \in I_{u_i} \\ 0, & otherwise\, if \quad i_j \notin I_{u_i} \end{cases} \tag{6}$$

As for the values of $p^c_{u_i,i_j}$, we illustrate the same example aforementioned, in a 5-starts scaled(S = 5) rating system for movies, as for the partial feature comes from item i_1 in the combined user's preference matrix,we get $p^5_{u_t,i_1} = 1$ and set the rest part as 0, thus we get vector $p_{u_t,i_1} = (1,0,0,0,0)$, the same process as for the rest part, we get $p_{u_t,i_3} = (0,1,0,0,0)$, $p_{u_t,i_4} = (0,0,1,0,0)$, $p_{u_t,i_5} = (0,0,0,1,0)$. For those items $i_{other} \notin I_{u_i}$ called missing value still can be derived according to (6). Finally, we combine each partial feature vector as a feature matrix

$$p_{u_t} = \left(p_{u_t,i_1}, \ p_{u_t,i_2}, p_{u_t,i_3}, \ p_{u_t,i_4}, \ p_{u_t,i_5}, \ \ldots, \ p_{u_t,i_n} \right)^T = \begin{pmatrix} 1,0,0,0,0 \\ 0,0,0,0,0 \\ 0,1,0,0,0 \\ 0,0,1,0,0 \\ 0,0,0,1,0 \\ \vdots \\ 0,0,0,0,0 \end{pmatrix}$$

Note that the value "0" is deferent from the binary indicator vector **0**. And we suppose $0 \times NULL = 0$, which will be useful in the next computation process of modified AE.

In this article, we modify the autoencoder to adapt the *NULL*-value (sparseness) scenario illustrated as above. The particular modifications are listed as follows.

1. For a particular user u_i, only activate the input units for the items rated by that user, and then feed those data to SDAE for encoding.
2. In steps of building each level AE of SDAE, when involving decoding the hidden layer in each local level DAE. We also only reconstruct the ratings for the items rated by the user u_i, which also means only backpropagate the error coming from rated items by the current user.c

In this modified model, it seems that every user in the system will have their own personal autoencoder based on their own collection of rated items. Each autoencoder only has a single training case. Weights and biases belonging to one user-specific autoencoder contribute to the entire global neural networks. Thus if the item was co-rated by many users, these users would share the corresponding weights for that item through the path from input to output in their own autoencoder networks. Figure 4 illustrate the modified AE model.

After all above preparation, then we can train the basic DAE and then stack them all to get SDAE. For each user u_i, we substitute p_{u_i} for x^i and carry out the feedforward step illustrated in Sect. 2, the final output $\hat{p}^c_{u_i i_j} \in [0,1]$ substituted for $\hat{x}^i$ represents the probability of item i_j rated in value c. Here we can regard c as a confidence level for

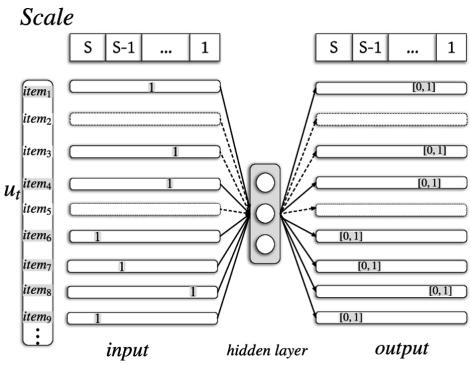

Fig. 4. Modified AE model in collaborative filtering

expressing not only the user rated the item but also indicate the preference extent the user showed when rating it.

By limiting the number of hidden neurons *d'* to a much smaller but reasonable scale than the total number of items *n*, the modified AE transform the sparse users behavior vector with null values into a dense feature vector. Thus decreasing the dimensionality of the user behavior vector space from *n* to *d'*, in which small perturbances inherited from noise in the data are eliminated, leaving only the strongest effects or robust dependency features. Consequently, it decreases the storage and computational requirements for the next collaborative filtering workflow.

3.2 Generate Recommended List and Prediction

After the workflow of SDAE, we can obtain the compact and efficient feature vector learnt from user's original preference behaviour. For a particular user u_i, we regularise the hidden feature vector h^{u_i} to $bicode_{u_i}$, the process of binarization given as follows, where 0.5 is the threshold value to decide if the hidden unit was highly activated by some latent features.

$$bicode_{u_i k} = \begin{cases} 1, & if\ h^{u_i} \geq 0.5 \\ 0, & otherwise \end{cases} \quad 1 \leq k \leq d' \tag{7}$$

We use the modified Hamming distance between these $bicode_{u_i}$ as the similarity metric formulated as

$$sim(u_1, u_2) = 1 - \frac{Hamm(bicode_{u_1}, bicode_{u_2})}{len(hidden\ units\ of\ final\ features)} \tag{8}$$

In the last step, we use the similarity between users to generate prediction ratings and recommendation list. Given the particular user u_i, and a certain item i_j, where $i_j \notin I_{u_i}$, we define $\deg_{u_i,i_j}$ to express the degree for item i_j worth of being recommended to user u_i.

$$\deg_{u_i,i_j} = \frac{\sum_{v \in Nbr(u_i,i_j)} sim(u,v)}{K} \tag{9}$$

where *Nbr(u,i)* are the neighbourhood users who have rated item i_j, and shared the most top-K similarity with the user u_i. The parameter K is the neighborhood size, needed to be optimized and determined experimentally. Then we can get the Top-N items in a descending order of $\deg_{u_i,i_j}$ as recommendation list for the target user.

We also define prediction ratings $score_{u_i,i_j}$ as

$$score_{u_i,i_j} = \bar{r}_{u_i} + \frac{\sum_{v \in Nbr(u_i,i_j)} (r_{v,i_j} - \bar{r}_v) sim(u,v)}{\sum_{v \in Nbr(u_i,i_j)} sim(u,v)} \tag{10}$$

4 Experiments and Discussion

4.1 Dataset and Evaluation

We use MovieLens datasets provided by GroupLens research team to evaluate the performance of our SDAE-CF approach and compare the results with several alternative baseline CF-based methods. To mimic different extent sparseness of the rating matrix, in this paper we experiment on MovieLens 100 k, 1 M, and 10 M datasets with their statistics summarised in Table 1.

Table 1. Statistics for data sets of the MovieLens

Type	Users	Movies	Ratings	Density%
100 k	943	1682	100,000	6.30
1 M	6040	3900	1,000,209	4.26
10 M	71567	10681	10,000,054	1.31

We randomly sample 80% ratings for each user as the training set and the rest 20% is used as the testing set. We generate five independent splits and report the averaged performance in our evaluations. We firstly take MAE as our evaluation metric to report experiments in user preference prediction. Then we use recall as another performance measure particularly for the Top-N recommendation task [13]. The evaluation rules are listed as follows, where $Test_u$ denotes the items having been rated by the user in the testing set. I_r is the set of Top-N recommendation generated from its training set denoted as $Train_u$.

$$MAE = \frac{\sum_{u \in U} |r_{i,j} - scrore_{u_i,i_j}|}{\sum_{u \in U} |Test_u|}, \; recall = \frac{\sum_{u \in U} |I_r \cap Test_u|}{\sum_{u \in U} |Test_u|}$$

4.2 Results and Discussion

In this section, we present the prediction quality of our SDAE-CF model. The benchmark methods include three conventional CF-based methods, that's the pure user-based method [2], the pure Item-based method [3], and the SVD feature [5].

In order to do a reasonable comparison, pre-settings of the experiments are as follows, according to [2], we set the neighbourhood size with 30 and use person correlation as similarity metric for user-based benchmark algorithm. As for [3], we choose the adjusted cosine as the similarity metric. The third baseline SVD feature method is a latent factor model intuitively having some analogies with our model, which leads us first thought of experiment on a middle sized dataset—MovieLens 1 M, to determine the optimal numbers of hidden units for our model, and simultaneously in line with the number of the latent factor in SVD. The sensitivity of MAE affected by the number of hidden units in a basic DAE is illustrated in Fig. 5, showing that 200 hidden units would be the choice for a better prediction performance and there is no obvious difference when the hidden units go beyond 200. Thus we set the number of latent factors in SVD feature with 200.

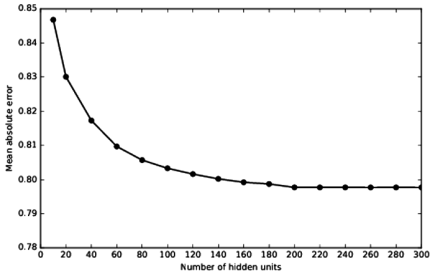

Fig. 5. The sensitivity of the number of hidden units

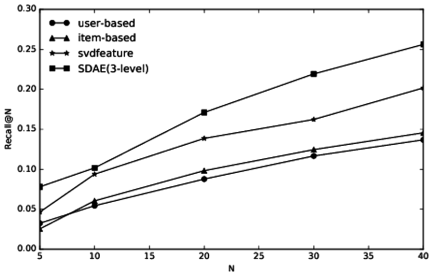

Fig. 6. Performance comparison by recall

Table 2 presents the MAE results from all three alternative baseline methods and our SDAE based model, in which fundamental structure also be involved. We can express the table from an empirical view that the DAE is relatively more accurate than the basic AE in faced with higher dimensional data. As it shows in Table 2, DAE and SDAE seems underperforming when coping with a lower dimension datasets, but still

Table 2. Performance on MovieLens datasets by MAE

CF-based method	ML-100 k	ML-1 M
Pure user-based	0.7489	0.7592
Pure item-based	0.7293	0.6983
SVD feature	0.7334	0.6944
AE (1 level)	0.7282	0.6965
DAE (1 level)	0.7482	0.6940
SDAE (3 level)	0.7262	0.6825

outperform other alternatives slightly. The purpose of introducing noise in autoencoder is for learning more robust features from the input. However, if the input itself are not strong enough, introducing noise would achieve nothing and may result in unrecoverable injuries to the original input. In general, AE based models are nearly the same level performance with the SVD feature methods by MAE.

The second form of interest prediction is Top-N recommendation task. We carry out experiments on MovieLens 10 M dataset to get the Top-N recommended items just according to (9), requires no further prediction score calculation. And we zoomed the range of N in [5, 10, 40]. Table 3 report the recall results in particular when the length of recommendation list is 20 and 40. Figure 6 illustrates that SDAE-based method outperforms the other three basic methods regarding recall metric, first also followed by the SVD feature. However, there is just a small performance gap between SDAE and SVD. In this respect, the two traditional neighborhood-based methods are in an inferior position, for in a sparser dataset the similarity calculation based on correlation is hindered by lacking enough co-rated items for users.

Table 3. Performance on MovieLens 10 M dataset by recall*

	N = 20	N = 40
CF-based method	Recall	Recall
Pure user-based	0.0877	0.1368
Pure item-based	0.0982	0.1456
SVD feature	0.1386	0.2017
SDAE (3 level)	0.1710	0.2561

5 Conclusion

In this paper, we analysis the problems of traditional collaborative filtering methods, then we proposed SDAE-CF model to alleviate the issues. On the one hand, using SDAE to encode the user preference vector break the limitation of similarity calculation among users depending on the common rated items, thus alleviate the loss of potential information. On the other hand, since the modified similarity based on Hamming distance can be calculated in constant time, decreasing the requirement for storing similarity extensively, which extends the scalability of traditional correlation-based algorithm. Experiments on three type scale MovieLens datasets show that SDAE-CF has potential in dealing with a high-dimension dataset and can achieve relative good performance in score prediction and Top-N recommendation task.

For the future work, we consider integrating with content-based techniques which can provide a more informational data foundation for extracting more useful latent features, and further to confirm the applicability of SDAE-CF model.

Acknowledgment. This work was supported by the National Nature Science Foundation of China (No. 61309013) and Chongqing Basic and frontier research projects (No. CSTC2014J CYJA40042).

References

1. Eppler, M.J., Mengis, J.: The concept of information overload: a review of literature from organization science, accounting, marketing, MIS, and related disciplines. Inf. Soc. **38**(5), 325–344 (2004)
2. Adomavicius, G., Tuzhilin, A.: Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 634–749 (2005)
3. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 230–237. ACM, Berkeley (1999)
4. Badrul S., George K., et al.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web (WWW 2001), Hong Kong (2001)
5. Yehuda, K., Robert, B., Chris, V.: Matrix factorization techniques for recommender systems. Comput. J. **42**(8), 30–37 (2009)
6. Chen, T., Zhang, W., Lu, Q., Chen, K., Zheng, Z., Yu, Y.: Svdfeature:a toolkit for feature-based collaborative filtering. JMLR **13**, 3619–3622 (2012)
7. Sarwar B., Karypis, G., Konstan, J., Riedl. J.: Application of dimensionality reduction in recommender systems—a case study. In: Proceedings of the ACM WebKDD Workshop, Boston (2000)
8. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**, 1527–1554 (2006)
9. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning, pp. 791–798. ACM, Corvalis (2007)
10. Georgiev, K., Nakov, P.: A non-iid framework for collaborative filtering with restricted Boltzmann machines. In: The 30th International Conference on Machine Learning (ICML 2013), pp. 1148–1156, Atlanta (2013)
11. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. JMLR **11**, 3371–3408 (2010)
12. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al.: Greedy layer-wise training of deep networks. Adv. Neural. Inf. Process. Syst. **19**, 153–160 (2007)
13. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the fourth ACM conference on Recommender systems. ACM, Barcelona (2010)